Optimal resource usage in ultra-low-power sensor interfaces through context- and resource-cost-aware machine learning

Steven Lauwereins[a], Komail Badami[a], Wannes Meert[b], Marian Verhelst[a]

[a]*KU Leuven ESAT-MICAS, Kasteelpark Arenberg 10, Heverlee B-3001 Belgium*
[b]*KU Leuven CS-DTAI, Celestijnenlaan 200A, Heverlee B-3001 Belgium*

Abstract

This paper introduces an approach that combines machine learning and adaptive hardware to improve the efficiency of ultra-low-power sensor interfaces. Adaptive feature extraction circuits are assisted by hardware embedded training to dynamically activate only the most relevant features. This selection is done in a context- and power cost-aware manner, through modification of the C4.5 algorithm. As proof-of-principle, a Voice Activity Detector illustrates the context-dependent relevance of features, demonstrating average circuit power savings of 70%, without accuracy loss. The RECAS database developed for experimenting with this context- and dynamic resource-cost-aware training is presented and made open-source for the research community.

Keywords: resource cost-aware classifier, context-aware machine learning, low-power sensor interface, adaptive circuits, power restricted classifier

1. Introduction

Interest in ubiquitous sensor networks is strongly increasing, spearheading applications relying on smart objects and smart environments. The sensors in these networks are expected to operate autonomously and continuously throughout their complete lifetime of multiple years. This restricts the average power budget of such sensors to a few μW, which is difficult to attain for current sensors [1, 2]. It is therefore important to discard irrelevant data as early as possible, in order

Email address: steven.lauwereins@esat.kuleuven.be (Steven Lauwereins)

to not waste scarce resources on the incoming sea of data. It has, for example, been well established that discarding irrelevant data before wireless transmission by on-board processing is far more power cost-effective compared to transmitting the raw data to a central data collecting node [3]. Taking this one step further, also within a sensor node, relevant features should be extracted as close to the raw sensor as possible to avoid power wastage.

At the same time, there is a trend within the hardware community to reduce the power consumption of their designs through the development of dynamically scalable and adaptive hardware [4, 5, 6, 7]. This kind of hardware can be reconfigured to the most power efficient configuration while meeting the requirements of the current use case. This use case is on average less stringent than the most demanding case, thus allowing for reduced power consumption. The variable behavior of the operational conditions and use cases is too complex and unpredictable to apprehend at design-time and is therefore best handled by autonomously exploiting this adaptivity. However, autonomously and optimally configuring these adaptive systems is still difficult. Current state-of-the-art (SoA) dynamic self-reconfigurable systems focus on optimizing power hungry hardware blocks such as DSPs for video compression and multi-core processors [8, 9]. Straightforward application of this dynamic self-reconfigurability paradigm is however impossible for power-scarce sensor nodes, due to the significant power overhead of the required embedded adaptivity management.

In previous work [10, 11, 12], we introduced and proved on silicon, a new sensing paradigm where hardware self-adaptivity is exploited within power-scarce sensor interfaces. Relevant information (features) are extracted as close to the raw sensor as possible to be power-efficient. The targeted sensor interfaces selectively extract features such that hardware resources of unselected features can be turned off to save power. This operating paradigm promises significant power savings (up to 10x) [10, 11, 12], however its implementation requires an advancement beyond the state-of-the-art of hardware self-adaptivity on three fronts.

Firstly, the dynamic control of feature selection and resulting hardware scalability needs to jointly take into account the impact of the scalability on the sensor's power budget, as well as on the sensor's classification accuracy. This results in a novel trade-off between power consumption and classifier accuracy, which can be exploited dynamically.

Secondly, the reconfigurability management has to be implemented with significantly reduced power overhead compared to the current SoA self-adaptive systems.

Thirdly, the configuration of the hardware needs to be optimal at every mo-

ment in time. However, environmental changes (from now on called context-switches) can lead to sub-optimal configuration of the overall system. This results in a need for at run-time detection of context-switches and a need for at run-time unsupervised retraining of the optimal configuration.

The here envisioned sensor interfaces overcome those challenges through the introduction of an adaptive feature extraction block dynamically configured with a low-cost machine learning scheme working together with a sporadically activated context anomaly detector (Fig. 1a). The analog feature extraction block allows dynamic deactivation of irrelevant features reducing the power consumption of this block. An additional power reduction is obtained by extracting the features in the analog domain because this requires a lower sampling frequency of the analog-digital-converter compared to digital feature extraction [12, 13]. A digital classifier uses the selected features for sensor signal classification with a target accuracy constraint. Embedded machine learning hardware is responsible for the optimal configuration of this feature extraction block and the definition of the classifier at any given moment in time. It therefore implements an unsupervised configuration management unit as well as a context anomaly detector which triggers retraining, Fig. 1a.

To reduce the overhead of this adaptivity management and the embedded classifier, the classifiers are realized using decision trees. Such trees enable power-efficient inference through the implementation of a decision tree classifier in dedicated hardware. Moreover, the embedded training of the decision tree inherently reveals the current importance of features, which is exploited to facilitate the dynamic feature selection process. Our previous work [10, 11, 12], showed that such low-cost machine learning is capable of controlling adaptive sensor node hardware resources within the sensor node's energy budget. This paper extends our previous work along four axes:

1. Presenting a detailed description of the algorithmic modifications to the C4.5 algorithm [14] made for the extension to true resource-cost-awareness.

2. Enabling optimal unsupervised training of a classifier under a predefined power budget through dynamic tracking of resource-selection within a best-first tree training scheme.

3. Enabling context-switch detection through one-class decision trees.

4. Release of our open-source context- and resource-cost-aware feature database (RECAS) for a voice activity detector, made available to the research community.

In this regard, Section 2 introduces the notion of context-aware feature activation, to dynamically activate current most relevant features. Section 3 further expands this idea to context-aware and dynamic resource-cost-aware classification for improved power efficiency, taking also resource circuit power cost into account. It discusses in depth the necessary algorithmic changes required for optimal training. Section 4 describes the method used to train decision trees in an unsupervised way and the method developed to detect context-switches. Section 5 applies the derived approach on a proof-of-principle hardware design of a Voice Activity Detector, demonstrating up to 10X reduction in power consumption or up to 10% increase in accuracy. Subsection 5.1 introduces our RECAS database allowing context- and dynamic resource-cost-aware simulations, made available to the research community in an open source scheme.

2. Context-aware feature selection

In many applications, the relative information content of a feature is highly context-dependent. Depending on the context, some features port a more distinctive value towards the classes of interest. Examples are acoustic classifiers, prone to various types of background noises, or patient-specific biomedical data classification [15].

To always operate the target sensing systems at maximal resource efficiency, feature selection should be done in a context-aware way. This context-aware feature selection allows for two different approaches. Firstly, by only activating discriminative features within the current operating context, the amount of extracted features and active hardware resources is dynamically adapted (see Fig. 1a). This reduces the overall power consumption (Fig. 1c) without losing classification accuracy. Alternatively, by activating all features, the context specific training of a tree makes it possible to increase the accuracy due to a tighter cluster of data points within each class.

Both approaches are illustrated in Fig. 2, which shows the ROC curves of a voice activity detection classifier trained and tested on signals with babble background noise. Fig. 2a shows the classification accuracy for a context-aware (CA) classifier with feature deactivation and for a context-unaware (CU) classifier. As previously mentioned the overall power consumption decreases, in this case with a factor 2, without losing classification accuracy. Fig. 2b shows the classification accuracy for a context-aware (CA) classifier trained on the correct context (solid line), for a context-aware classifier trained on the wrong context (dotted line) and for a classifier trained on all contexts, i.e. a context-unaware (CU) clas-

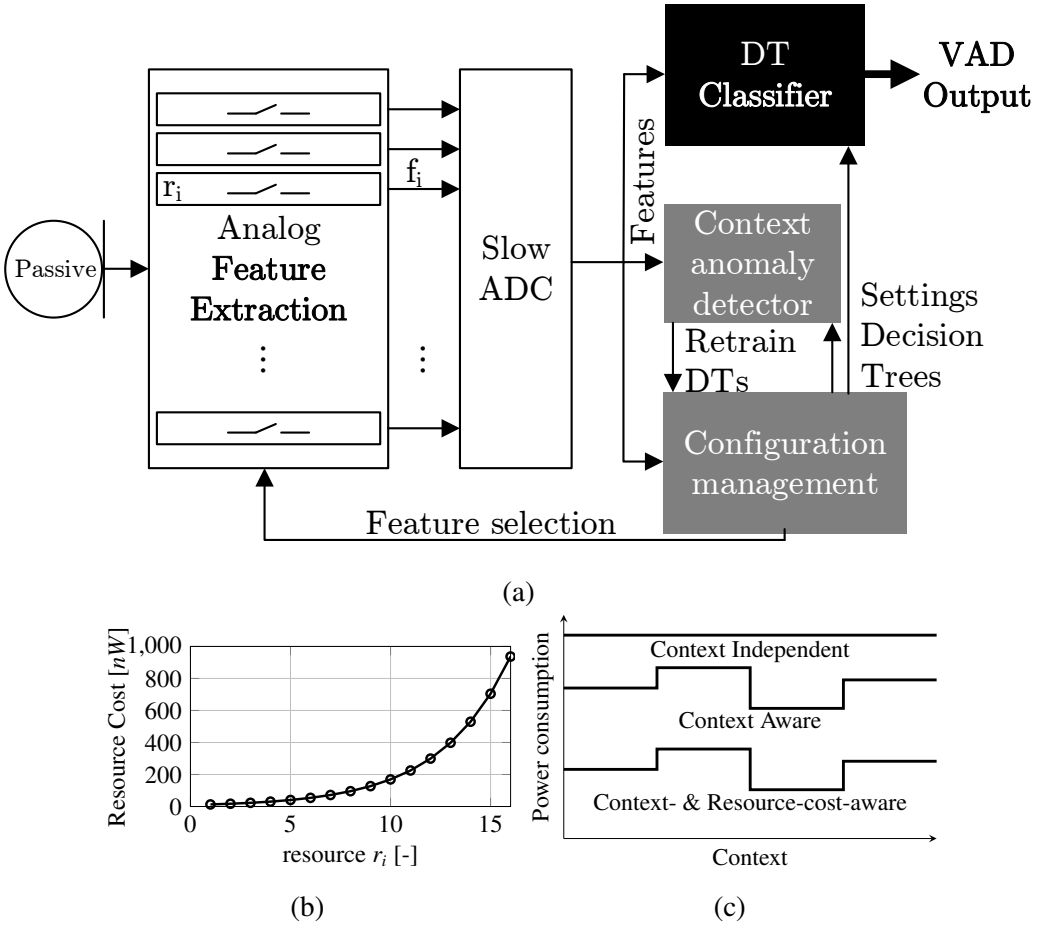


Figure 1: (a) Context- and Feature cost-aware classifier showing integration of adaptive hardware and machine learning. The grey blocks are only sporadically activated. (b) Illustrates the varying power cost across feature extraction hardware blocks. (c) Show anticipated power consumption versus context for three types of classifiers. (All applicable for acoustic sensing scenario introduced in section 5.)

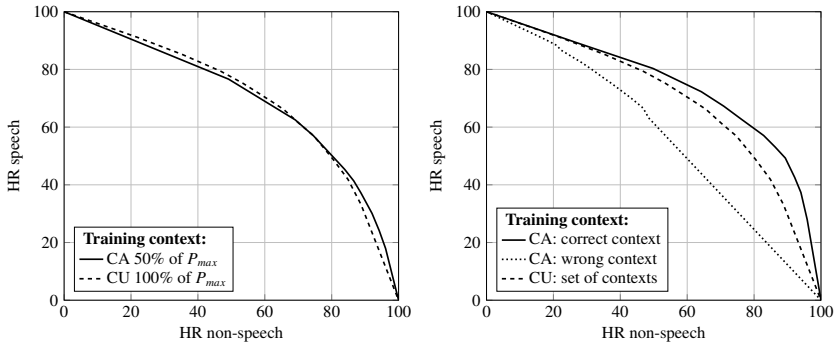


Figure 2: Comparison of the voice activity detection accuracy of a babble noise test case for a context-unaware classifier (CU) (equal combination of 8 different noise contexts) with (a) context-aware (CA) inference with feature deactivation, (b) context-aware inference without feature deactivation for the correct context (babble noise) and the wrong context (exhibition noise).

sifier (dashed line). As can be expected, accuracy of a classifier trained on a correct context-restricted dataset (e.g. babble noise) significantly outperforms the context-unaware classifier. A wrongly trained classifier, i.e. a classifier trained on the wrong context, performs poorly. This observation introduces a new trade-off when training the model, classification accuracy can be exchanged for lower power consumption.

In this work, decision tree based classifiers will be trained for different contexts. Within every context, the features used in the different nodes of the tree determine the discriminative features requiring activation. As will be demonstrated in Section 5, this context-awareness allows cutting the number of actively observed features by a factor of $\sim 1.6X$ in typical sensing applications. The classifying decision tree automatically orders these features in order of decreasing information gain, allowing dynamically trading-off the classification accuracy with the power consumption by preserving more or less nodes in the tree.

The specialization of a tree towards a specific context increases the sensitivity of context-switches (i.e. environmental changes) on the classification performance. In Fig. 2b, a wrongly trained classifier (dotted line) performs worse than a context-unaware classifier (dashed line), demonstrating this increased sensitivity on context-switches. It is therefore important to timely detect such context-switches, in order to retrain the system on the new context, hence maintaining the high classification quality. The background noise context can be monitored by sporadically activating an unsupervised context anomaly detector (Fig. 1a), which

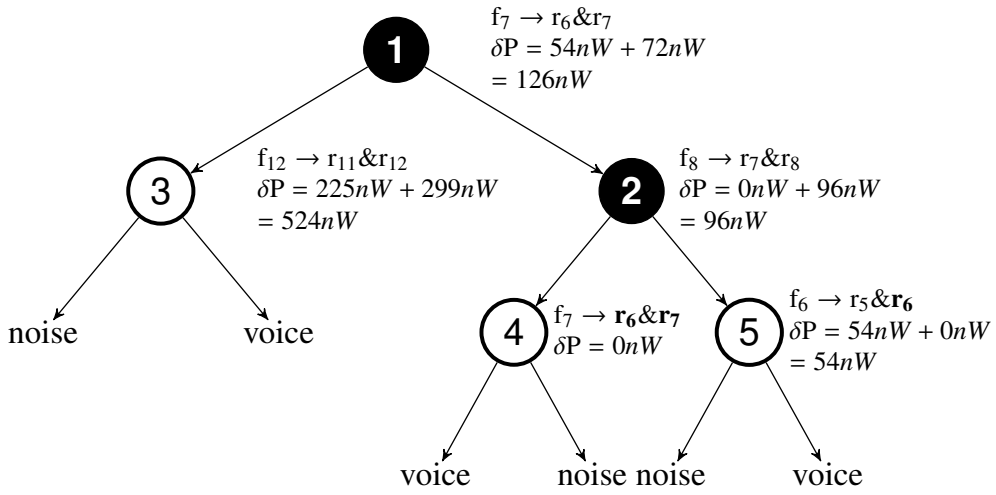


Figure 3: Example of a best-first tree construction scheme. The black nodes represent existing nodes, while the white nodes represent nodes under consideration. Resources r_i used in existing nodes because of their feature f_i do not contribute to the additional power consumption δP of the nodes under consideration, and are therefore written in bold.

classifies the background noise as the current context or as an anomaly. Whenever an anomaly is detected (i.e. context-switch), the VAD classifier will be retrained. This context anomaly detector is described in section 4.

3. Cost- and Context-aware feature selection

Within a given operating context, the model building block of Fig. 1a has to construct a decision tree that achieves the target accuracy with minimal hardware resources. However, in contrast to our previous work, the mapping between hardware resource usage and feature selection is not always linear. The decision tree training and feature selection should therefore be expanded with knowledge of the actual feature cost. This section will, to this end, introduce dynamic resource-cost-aware feature selection. This is achieved through three algorithmic changes to the popular decision tree training algorithm C4.5 [14].

3.1. Resource cost

In adaptive ultra-low-power chips, features (f_i) are extracted by combining a limited number of shared hardware resources (r_i). Moreover, hardware resources do not all demand the same amount of power, e.g. Fig. 1b. The additional power consumption required to add a feature is therefore dependent on the resources

that are already activated due to features that are already selected, and on the resource-specific power consumption. The added power consumption of an additional feature is thus a dynamic variable while building the tree. Fig. 3 illustrates this dynamic additional power consumption of a feature during tree training. The decision tree shown has two existing nodes (black) and three nodes under consideration (white). The additional power consumption required for node 4 is $0nW$ because it requires resources that were already needed for node 1, namely resources 6 and 7. These resources were already activated and don't require extra power consumption when being reused. However, the original C4.5 decision tree algorithm trains the model only based on the information gain of the feature f_i and is completely oblivious to its accompanying power cost.

While state-of-the-art research, such as [16], introduced cost-aware training for decision trees, cost optimization is primarily focused on misclassification cost, while classification cost reduction through feature deactivation is less researched. A popular technique that does minimize cost only extracts the features needed while traversing the tree during classification [17]. This is not transferable to embedded classification in sensor interfaces, as it requires sequential feature extraction, which is impossible with time averaging features. Additionally, both methods are currently limited to discrete attributes and do not yet support continuous attributes. More work is hence needed for very lower power applications to dynamically sensitize the machine learning algorithm to the feature-specific or even resource-specific power cost leading to improved power consumption profiles as shown in Fig. 1c.

In this work we present a decision tree algorithm that includes the resource-specific power-usage to ensure a bias towards features with an optimal trade-off between cost and gain. This resource-specific context-awareness differs from the previously implemented feature-specific context-awareness [10] in the way it calculates additional power consumption for an added feature. It dynamically takes the already used resources into account while feature-specific context-awareness uses a static fixed power consumption for every feature. The proposed algorithm, Alg. 1, therefore uses a splitting criterion that both considers a feature's discriminative nature and its additional resource-specific power-cost:

$$Split_{(n,f_k)} = \frac{IG_{(n,f_k)}}{(1-\alpha)\cdot \delta P_{(n,f_k)} + \alpha \cdot P_{(n-1)}} \tag{1}$$

with $IG_{(n,f_k)}$ the information gain [14] of feature f_k at build step n, α a weight factor, $\delta P_{(n,f_k)}$ the additional power consumption caused by the addition of feature f_k in build step n taking all resources r_i into account that were activated by feature

selection in the $n-1$ previous build steps. As an example Fig. 3 illustrates this, already used resources $\mathbf{r_i}$ required for nodes under consideration during tree construction (white) are shown in bold and do not require any additional power. $P_{(n-1)}$ represents the total power consumption of the built tree up to build step $(n-1)$. The recursive form of this equation is used in Alg. 1 line 29. This splitting criterion encourages re-use of features, because $\delta P_{(n,f_k)} = 0$ when feature f_k was already used higher up in the tree, as can be seen in Fig. 3 node 4. The additional power consumption $\delta P_{(n,f_k)}$ for a feature f_k at step n will be smaller when it can reuse already activated hardware resources, Fig. 3 node 5. This formulation of cost-aware feature selection can thus be called *dynamic resource-cost-aware* feature selection instead of *feature-cost-aware* feature selection. The formulation of this equation stresses power consumption more in the beginning of the tree construction, when $P_{(n-1)}$ is still small. This ensures cheaper features at the top of the tree which improves the scalability of the total power consumption. In other words, this dynamic resource-cost-aware feature selection modification gives "best value for money" or highest resource efficiency.

3.2. Adaptive power limitation

Many applications have use cases where the power budget varies dynamically from their nominal use case [18, 19]. Examples are audio sensors without activity in their environment for extremely long periods or cellphones where the battery is almost dead. During those periods of power scarcity, those use cases often also have less strict accuracy requirements than in the nominal use case. A sensor should then limit its power consumption to the maximal allowed power while maximizing accuracy. The proposed modifications to the C4.5 algorithm enable this flexibility, as it allows the specification of a maximum overall power cost to train trees within the power limits. The algorithm then only selects features that keep the total power consumption below the allowed value, Alg. 1 line 30. This approach maximizes the classification accuracy for the given maximum power consumption, since the limitation is not on the tree size but on the used hardware resources. This is not the case when the power limitation would be applied when pruning the tree, i.e. after tree construction. The latter would limit the tree size and thus reduce the achieved accuracy.

3.3. Best-first tree construction

In the original C4.5 algorithm the subtree of a sibling has no influence on the splitting criterion in the current node. Therefore, the order in which the tree is built is irrelevant. However the introduction of dynamic resource-cost-aware feature

Algorithm 1 Resource-cost-aware modifications to C4.5: resource-cost-aware feature-selection, adaptive power limitation, best-first tree construction

```
1: function buildTree(resourcePowerMapping, featureResourceMapping, maxP_Tree)
2:     P_Tree ← 0
3:     usedResources ← 0
4:     list currentLeafs ← leaf with the whole training set
5:     while !finished do
6:         bestNode ← 0
7:         for all currentLeafs do
8:             newNode ← createNewNode(usedResources, resourcePowerMapping, featureResourceMapping, P_Tree)
9:             if newNode > bestNode then
10:                bestNode ← newNode
11:            end if
12:        end for
13:        update tree with bestNode
14:        update list currentLeafs
15:        update usedResources
16:        P_Tree ← P_Tree + δP_bestNode
17:        if no bestNode then
18:            finished ← true
19:        end if
20:    end while
21: end function
```

22: **function** CREATENEWNODE(*usedResources*, *resourcePowerMapping*, *featureResourceMapping*, P_{Tree})
23: $bestSplit \leftarrow 0$
24: $newNode \leftarrow 0$
25: $bestFeature \leftarrow 0$
26: **for all** features f_i **do**
27: $resourcesToActivate \leftarrow featureResourceMapping(f_i)$ **&** **!***usedResources*
28: $\delta P \leftarrow \sum resourcePowerMapping(resourcesToActivate)$
29: $Split = \frac{IG}{(1-\alpha)\cdot\delta P+\alpha\cdot P_{Tree}}$
30: **if** $\delta P + P_{Tree} < maxP_{Tree}$ **&** $Split > bestSplit$ **then**
31: $bestFeature \leftarrow f_i$
32: **end if**
33: **end for**
34: $newNode \leftarrow$ create $Node(bestFeature)$
35: **return** *newNode*
36: **end function**

selection introduces dependencies between siblings in the tree. Every new node potentially changes the power consumption of the entire tree and thus changes the cost of the features. Since the new split criterion, Eq. 1, takes the additional power consumption of a feature and the current power consumption of the tree into account the order of node creation matters to the overall achieved accuracy. Especially if pruning is performed to reduce over-fitting or to reduce the power consumption after the tree is build. The importance of a node increases towards the top of a tree and it is therefore important to select the feature that gives "the best value for money". The adaptation of the C4.5 algorithm, introduced in this paper, therefore builds trees based on a best-first construction scheme instead of the typical depth first scheme. The algorithm calculates a node for every leaf that the tree under construction currently has (white nodes in fig. 3) to evaluate which leaf is best replaced by a node, Alg. 1 lines 7-12. This node is then added to the tree followed by an update of the list of current leafs, the list of used resources and the total power consumption of the tree. The current leafs list is initialized with a leaf containing the whole training set.

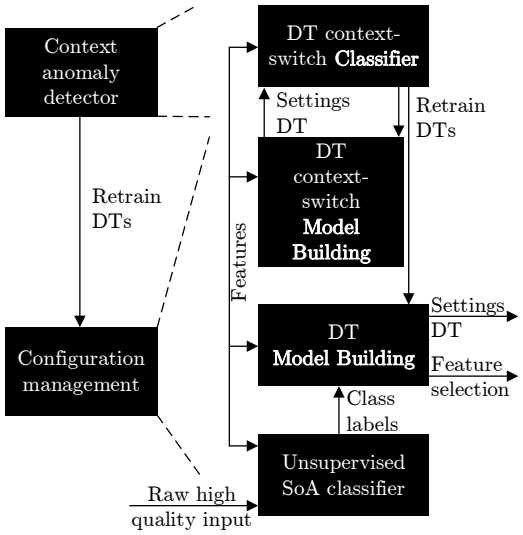


Figure 4: Block diagram of unsupervised configuration management and context anomaly detection.

4. Unsupervised configuration management and context-switch detection

To implement hardware self-adaptivity within power-scarce sensor interfaces, the hardware has to configure itself autonomously into the current optimal configuration. This requires two major control steps. Firstly, the system needs to detect when the optimal configuration deviates from the current configuration, which happens when the behavior of the context changes (i.e. context-switch). Secondly, when the system is not anymore in its optimal configuration, it has to retrain what the optimal configuration is and tune itself to this configuration. The sensor interface system proposed in this paper solves both control steps through two sporadically activated system blocks. The first step is taken care of by a context-switch detector (i.e. context anomaly detector) (section 4.1), while the second step is handled by the configuration management block (section 4.2), Fig. 4.

4.1. Context-switch Detection

The here proposed context-switch detection consists of a a context-switch model building block that produces an efficient context-switch classifier (top Fig. 4). The model building step trains a decision tree that differentiates between the current context and a new, undefined context. The context-switch decision tree classifier makes use of the mean and standard deviation of the original features over a number of successive frames. These statistical features are defined as: $f' =$

$\mu(f_{k,n} : f_{k,n+m})$ and $\sigma(f_{k,n} : f_{k,n+m})$, where μ is the mean, σ the standard deviation, and $f_{k,n+m}$ is feature k in frame $n + m$, with n a frame in the recorded frame-set of the currently observed context and m the number of successive frames used to compute one statistical feature. The training samples for the currently observed context is expanded with uniformly distributed random values lying between $[-2 * (\mu(f') - min(f')) + \mu(f')]$ and $[2 * (max(f') - \mu(f')) + \mu(f')]$.

As the decision tree algorithm requires at least two classes for decision tree training, also a training set for the anomaly class is needed. A common approach followed by anomaly detection algorithms to solve this problem is to produce this anomaly class that represents all other possible contexts artificially [20, 21]. To create such a *world class* our approach first extracts the statistical features for the currently observed context, namely the statistical distributions of the mean and standard deviation of every feature over a number of successive frames. It then constructs the world class by creating a set of samples with the inverse feature space of the currently observed context. This is done by generating random values from $-9\sigma' + \mu'$ to $9\sigma' + \mu'$ with a probability of $1 - \mathcal{N}(\mu', \sigma')$, where μ' and σ' are the mean and standard deviation of the previously described distribution, Fig. 5. The wider Gaussian for the world class is chosen to allow for uncertainties on the distributions of the known class. With the sample set recorded from the currently observed context and the created world class set, a decision tree is then trained using the previously described adaptation of the C4.5 algorithm. The context-switch classifier block is sporadically activated (e.g. during $5s$ every $30s$) to infer if retraining of the whole system is needed. It is supplied with the mean and standard deviation of the extracted features over a restricted set of successive frames. The possible activation of extra features creates only a small overhead to the overall power consumption because of the sporadic activation of the context-switch detection.

4.2. Unsupervised configuration management

To autonomously reconfigure the system upon context-switch to the optimal configuration, the system needs to retrain itself in an unsupervised way. However, decision trees require labeled training sets, thus missing the requirement of unsupervised training. The here proposed sensor interface architecture solves this by using a power hungry unsupervised SoA classifier to label the incoming data during training (bottom Fig. 4). This principle of two-stage training where the first step uses a complex but powerful classifier and the second stage uses the classification information from the first step to train a more compact and efficient classifier is called model compression [22]. The training set is then annotated with

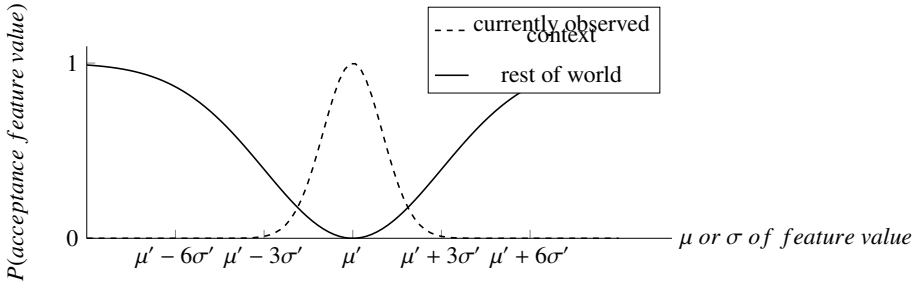


Figure 5: Probability of acceptance of a feature value into the feature space of a currently observed context or the rest of the world.

these labels to form the input of the decision tree training phase. This method introduces mistakes in the labeling of the training set compared with supervised training. However section 5.5 will show that the loss in accuracy caused by the pre-labeling is negligible. The compute intensive unsupervised SoA classifier has an order of magnitude higher power consumption than the trained decision tree and its feature extraction and runs on a DSP. The resulting high power consumption of the training phase is allowable since the phase will only be sporadically activated (worst case once a minute with a training duration of 5s).

5. Proof-of-principle - Voice Activity Detector

This section will apply the proposed approach to the design of a relevant power-scarce sensor interface to illustrate its merits in a realistic design. An acoustic interface for Voice Activity Detection (VAD) in mobile devices, targeting continuous voice versus noise classification under various background noises (contexts), is extremely power-constrained due to its always-on operation. This design will serve as a proof-of-principle for unsupervised decision tree training (section 5.5), for context-aware (section 5.3) as well as context- and dynamic resource-cost-aware (section 5.4) voice/noise classification and for context-switch detection (section 5.6). Accuracy gains of more than 10% and power savings from 2X up to 10X over context independent operation will be demonstrated. We will moreover open up our context- and resource-cost-aware speech (RECAS) database in an open-source format.

5.1. Data generation

The input of the analog feature extraction block (Fig. 1a) is an audio stream constructed from voice and noise samples from the widely used NOIZEUS database

[23]. The voice samples were scaled to achieve the different SNRs and were then added to each noise stream. The average power of these noise streams was equalized to each other for inter-noise comparability. Features are then, in the analog domain, extracted out of this audio signal by decomposing it into 16 frequency bands spaced on a logarithmic scale. A feature f_k is defined as the energy difference between neighboring mel-shaped frequency bands over a time period of $20ms$ and with frame shifts of $10ms$:

$$f_k = E(k) - E(k-1) \tag{2}$$

with $k \in [1; 16]$ and $E(k)$ the sensed energy in frequency band k, over a time frame extracted by an analog resource:

$$E(k) = \overline{|butter_{(f_l(k),f_h(k))}(y)|} \tag{3}$$

A first order butterworth filter (*butter*) is used with frequency limits $f_l(k) = 30Hz \cdot 1.33^{k-1}$, $f_h(k) = 30Hz \cdot 1.33^k$ and y the amplified signal of the passive microphone in a frame. This amplified signal is substituted by the audio signals created from the NOIZEUS database. Every energy value $E(k)$ is independently extracted in the analog domain through an analog filter stage, which we will denote as a basic analog resource r_i with a particular power consumption cost, Fig. 1b.

To create a ground-truth, every acoustic frame is labeled as voice or noise by running the SoA voice-activity detector of Ramírez [24] over the noise free voice files and those labels are combined with the corresponding feature vector. For supervised training, the training sets are labeled with the ground-truth without hangover scheme while the test sets are labeled with a hangover scheme of $120ms$. For real life applicability, we have also augmented our system with unsupervised training, as described in section 4. Therefore, noise corrupted training sets are labeled with the VAD of Ramírez without hangover scheme, while the test sets use the ground-truth previously described with a hangover scheme of $120ms$. Such a hangover scheme delays the voice-noise transition and thus cancels out silences between words in a sentence or labels silent vowels (e.g. “s”) at the end of a word as voice. The acoustic streams thus classify voice activity at a sentence granularity. Additionally, we constructed a matrix, linking every single feature to its resource usage. Finally, the actual power cost of every resource is captured in a power cost vector. This power consumption is determined by executing analog circuit simulations of the filter bank in $90nm$ CMOS, revealing strongly rising resource power cost for increasing frequency band numbers, as shown in Fig. 1b.

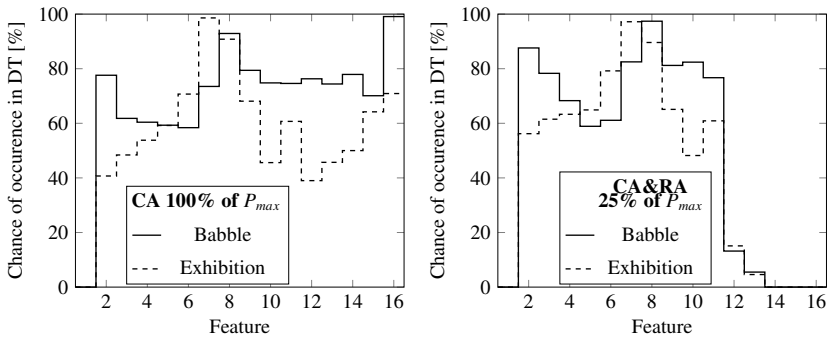
In the current SoA research, there is, to the best of the authors knowledge, no dataset available containing feature vectors linked to the power consumption of the hardware necessary for this feature extraction. Such a database is however indispensable to spearhead research on energy-awareness in classifying sensor interfaces and more generally, on cost-aware training. This paper therefore introduces an open-source context- and resource-cost-aware speech (RECAS) database containing of such data.[1] It consists of the labeled acoustic streams of the previously described features for eight different noise contexts, in both a training set configuration (no hangover scheme applied) and a test set configuration (120*ms* hangover scheme applied). It also contains the previously mentioned feature-resource mapping matrix and the simulated resource-specific power numbers for all the used analog resources, allowing context- and resource-cost-aware simulations.

5.2. Measurement setup

All results reported in this paper have been conducted using our RECAS database, which is based on the widely used NOIZEUS dataset [23], except for context-switch detection, which is tested on the DEMAND database [25]. Noise and voice samples are combined for varying signal-to-background-noise ratio (SNR) values and context, using MATLAB. To ensure real life applicability, one common model is trained across a SNR-range, spanning from 0dB to +5dB. A balanced dataset of 250 samples each for voice and noise is used as training set, while the testing set consisted out of 15.930 samples equally divided between voice and noise. The classifier is thus trained for 5*s* and then tested for 159.30*s*. This corresponds to the real life asymmetry desired for an energy efficient VAD. Decision tree training and evaluation is done deploying the modified C4.5 algorithm (based on the J48 implementation in Weka). Table 1 shows the other specifics of the experimental setup for the DTs. The ROC curves of the DTs are made by varying the penalty for speech misclassification from 1 to 3 and for non-speech from 1 to 30 and every simulation point is the average over 1000 DTs. To achieve a range of SNRs for training and testing, the amplitude of the voice files is scaled with regard to the average noise and voice power. The average noise power of different noise contexts is made equal to allow for a fair comparison between different noises.

Context-switch detection uses statistical features derived from the energy features used by the VAD. It uses the mean and standard deviation of 500 successive frames for every energy feature of the VAD. This allows for accurate description

[1] Available at: `http://www.esat.kuleuven.be/CubiqLab/recas`



(a) (b)

Figure 6: Chance of occurrence of a feature in a trained decision tree (DT) for context-aware (CA) DTs with the analog FE block using 100% of its maximal power consumption (a), and context- and dynamic resource-cost-aware (CA&RA) DTs with a power restriction of 25% on the FE block (b), under babble noise and exhibition noise. CA&RA DTs use less expensive features than CA DTs, thus reducing the FE power consumption.

of the current context. The training sets used in the performance discussion of the context-switch detection are 2.000 samples long, extended with 2.000 virtual samples for the current context (total training time of 25s) and 56.005 samples for the description of the world, created as described in section 4. The test sets are for every tested context 20.000 samples long. The penalty for misclassification of the currently observed context versus the world was set equally large. Also in this case, decision tree training and evaluation is done deploying the modified C4.5 algorithm in Weka.

5.3. Context-aware voice/noise classifier

To demonstrate the benefits of context-awareness, a voice/noise classifier is trained in a supervised way. Fig. 6a shows that context-aware feature selection only activates a subset of the features in a context-specific manner. Different features are discriminative in different contexts (dashed line versus solid line).

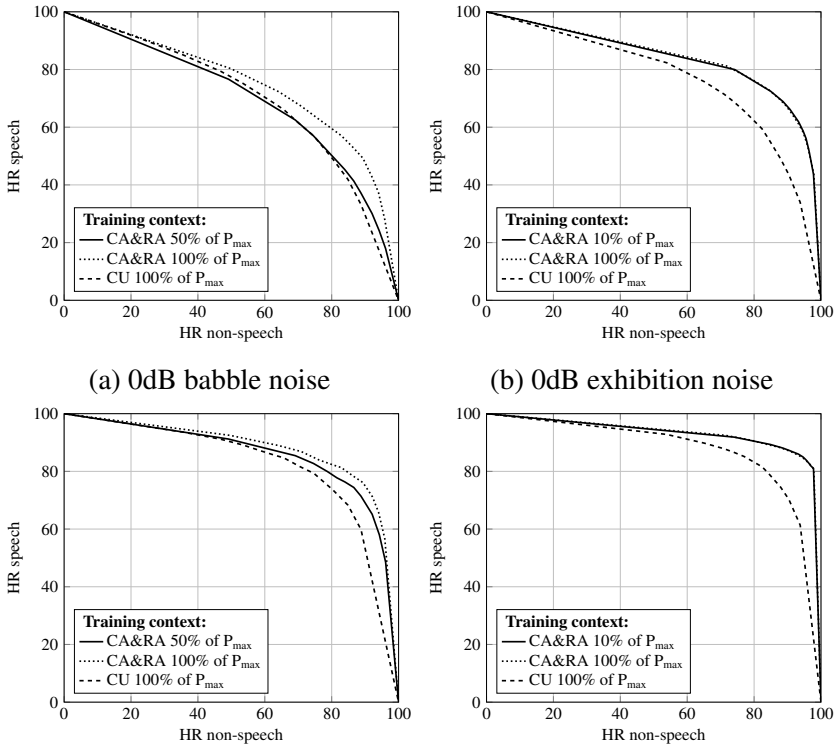
Table 1: Values of parameters used in the implementation of the proposed algorithm, for a sampling rate of 8kHz

frame shift:	10ms	frame length:	20ms		
hangover:	120ms	training SNR:	0-5dB	α:	0.75
#training frames:	500	#test frames:	15930		

Simulations show that only activating discriminative features allows on average a 1.6X reduction in activated hardware resources. E.g. feature 16 is always selected in a context-aware classifier for babble noise (Fig. 6a solid line), while it is less often selected in context-aware decision trees trained on exhibition noise (Fig. 6a dashed line). Moreover, this kind of feature selection allows for more than 5% accuracy increase compared to a context independent scenario where all contexts are trained simultaneously. This accuracy gain is visible in Fig. 7 by comparing the dotted lines (context-aware feature selection without power restriction) with the dashed line (context independent feature selection without power restriction). The accuracy gain of context-aware versus context independent trees increases with increased SNR for the most difficult kind of contexts, namely time varying contexts, such as babble noise. Of course, context detection adds a cost penalty but contexts usually do not change very often and is therefore not as expensive as voice recognition which runs continuously [26]. But for a cell phone this e.g. can be achieved by combining observed cellular IDs and inertial sensors [27]. We suspect that changes in context only happen at worst once a minute. This happens for example when someone walks out of a building on to the street and then enters a taxi. However, the average time between context-switches occurs less often, people sit for long time periods at home, work, in a car, etc. In the worst-case situation of one context-switch per minute, all resources are activated 20% of the time increasing overall power consumption with a factor 2. However in a nominal day, context switches occur less often, expected less than 1% of the time increasing overall power consumption only slightly [12].

5.4. Dynamic resource cost- and context-aware voice/noise classifier

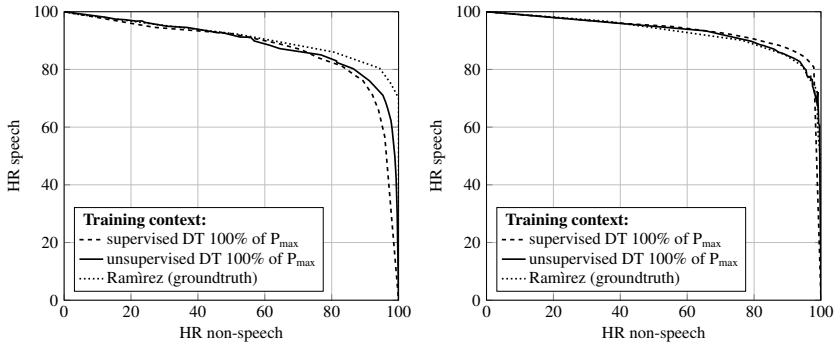
Dynamic resource-cost-awareness bases classification on information gain per resource cost rather than information-gain alone (Section 3). The impact of this modification (Fig. 6b) compared to pure context-aware training (Fig. 6a) can be seen in Fig. 6. Context-aware and dynamic resource-cost-aware feature selection prefers resources with relatively low frequency bands for voice/noise classification since these frequency bands demand lower power. Depending on the noise context, this leads to power savings from a factor 2 up to 10 for the same accuracy compared to context-unaware (CU) tree creation, as can be seen in Fig. 7 by comparing the solid lines with the dashed lines. Furthermore, accuracy with respect to context-unawareness is boosted when context-awareness and dynamic resource-cost-awareness is applied without any power limitations (dotted line in Fig. 7) Fig. 7 also shows that stationary noise such as exhibition noise is accurately classifiable with extremely power restricted context- and dynamic resource-cost-aware



(a) 0dB babble noise (b) 0dB exhibition noise

(c) 10dB babble noise (d) 10dB exhibition noise

Figure 7: ROC curves showing classification accuracy for Context/Cost (un)aware scenarios under various contexts and SNRs. Context- and dynamic resource-cost-aware feature selection without power restrictions (CA&RA 100% of maximum resource-power consumption) achieves higher accuracy over all test cases, while CA&RA feature selection with a resource-power restriction of 50% of the maximum performs as good or better than the context-unaware (CU) test case.



(a) 10dB babble noise (b) 10dB exhibition noise

Figure 8: ROC curves showing classification accuracy for supervised training (dashed line), the classifier used for ground-truth during unsupervised training (dotted line) and unsupervised training through model compression (solid line).

decision trees. The power consumption of the feature extraction of such trees can be reduced 10 fold without any noticeable accuracy loss. However, less extreme power savings are possible for time varying noises such as babble noise, which still achieves the same accuracy as the context-unaware classifier even though the power consumption is halved.

5.5. Unsupervised classifier training

Unsupervised training of decision trees is achieved by using a power hungry unsupervised SoA VAD to label the training set. The usage of a power hungry VAD increases the power overhead during training, however training happens only sporadically (less then once a minute) which makes the impact on the overall power consumption negligible [12]. It is expected that the classification mistakes of the SoA VAD influence the accuracy of the trained decision trees. Simulations show however that this accuracy drop is very small (Fig. 8). When the supervised DT outperforms the SoA VAD used as labeler, the unsupervised DT performs as good as the labeler. When decision trees themselves become the limiting factor, unsupervised training performs as good as the supervised training. This means that the trained decision trees are capable of accurately modeling the power hungry unsupervised SoA VAD, which proves that model compression is an effective technique to reduce power consumption while maintaining accuracy and independence.

Table 2: Accuracy results for context-switch detection through an anomaly decision tree consuming only 10% of maximal feature power consumption.

		Test context		
		Restaurant	Car	Square
Training context	Restaurant	84%	100%	99%
	Car	84%	80%	73%
	Square	82%	100%	81%

5.6. Context-switch detection

The final step for context-awareness in our VAD system is to sporadically check for context-switches, because these context-switches require retraining of the VAD for optimal accuracy performance. To guarantee accuracy at all time it is preferable to detect every context-switch, sacrificing energy by unnecessary retraining for unchanged contexts. Our approach explained in section 4 has this bias towards context-switch detection. Table 2 shows the accuracy for context-switches (outside of positive diagonal) and for unchanged contexts (positive diagonal) with no additional resources to be activated. The worst case accuracy of 73% for context-switch detection and of 80% for unchanged contexts are sufficient to support VAD retraining within the allowed 5% power overhead. The feature extraction power overhead for context-switch detection is therefore non-existent.

6. Conclusions & future work

This paper enables hardware-embedded machine training to smartly control power-aware adaptivity in future sensor interfaces. Through the introduction of dynamic resource-cost-aware feature selection and unsupervised decision tree construction, hardware resource efficiency is maximized by controlled feature (de)activation. To enable optimal configuration of the system context-switches are automatically detected allowing for retraining when the context (i.e. environmental condition) has changed. A VAD proof-of-principle demonstrates up to 10X power savings in real sensor interfaces, a gain invaluable towards ubiquitous sensing in the internet-of-things. This proof-of-principle also demonstrates unsupervised training of decision trees while maintaining SoA classification as well as accurate context-switch detection. The here presented paradigm opens up numerous other acoustic event detection applications, ranging far beyond VAD, and can also be ported to other ultra-low-power sensor interfaces, such as gesture recognition. Furthermore, the paper introduced a novel open-source context- and resource-cost-aware

dataset, allowing researchers to further develop and test power optimization algorithms and systems.

Acknowledgments

Steven Lauwereins is funded by a Belgian FWO Grant.

Bibliography

[1] N. Van Helleputte, A multi-parameter signal-acquisition SoC for connected personal health applications, in: International Solid-State Circuits Conference, 2014, pp. 314–316.

[2] P. Jourand, R. Puers, The BladderPill: An in-body system logging bladder pressure, Sens. Actuators A: Phys. 162 (2) (2010) 160–166.

[3] B. Calhoun, J. Lach, Body sensor networks: A holistic approach from silicon to users, Proc. IEEE 100 (1) (2012) 91–106.

[4] M. Ashouei, J. Hulzin, J. Van Ginderdeuren, A voltage-scalable biomedical signal processor running ECG using 13pJ/cycle at 1MHz and 0.4 V, in: International Solid-State Circuits Conference, 2011, pp. 332–334.

[5] H. Kim, S. Kim, N. Van Helleputte, A. Artes, M. Konijnenburg, J. Huisken, C. Van Hoof, R. F. Yazicioglu, A Configurable and Low-Power Mixed Signal SoC for Portable ECG Monitoring Applications, IEEE transactions on biomedical circuits and systems 8 (2) (2014) 257–67.

[6] M. Konijnenburg, Y. Cho, Reliable and energy-efficient 1MHz 0.4 V dynamically reconfigurable SoC for ExG applications in 40nm LP CMOS, in: International Solid-State Circuits Conference, 2013, pp. 430–432.

[7] M. Verhelst, N. Van Helleputte, A reconfigurable, 0.13m CMOS 110pJ/pulse, fully integrated IR-UWB receiver for communication and sub-cm ranging, in: International Solid-State Circuits Conference, 2009, pp. 250–252.

[8] H. Hoffmann, J. Holt, G. Kurian, Self-aware computing in the Angstrom processor, in: Design Automation Conference, ACM Press, New York, NY, USA, 2012, p. 259.

[9] A. Bernauer, O. Bringmann, W. Rosenstiel, Generic Self-Adaptation to Reduce Design Effort for System-on-Chip, in: IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Ieee, San Francisco, CA, USA, 2009, pp. 126–135.

[10] S. Lauwereins, K. Badami, W. Meert, M. Verhelst, Context- and cost-aware feature selection in ultra-low-power sensor interfaces, in: ESANN, 2014, pp. 93–98.

[11] S. Lauwereins, W. Meert, J. Gemmeke, M. Verhelst, Ultra-Low-Power Voice-Activity-Detector Through Context- and Resource-Cost-Aware Feature Selection in Decision Trees, in: MLSP, 2014, p. 70.

[12] K. Badami, S. Lauwereins, W. Meert, M. Verhelst, Context-Aware Hierarchical Information-Sensing in a 6 W 90nm CMOS Voice Activity Detector, in: International Solid-State Circuits Conference, 2015, pp. 430–431.

[13] A. Raychowdhury, A 2.3 nJ/Frame Voice Activity Detector-Based Audio Front-End for Context-Aware System-On-Chip Applications in 32-nm CMOS, Solid-State Circuits, IEEE Journal of 48 (8) (2013) 1963–1969.

[14] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, CA, USA, 1993.

[15] J. Yoo, L. Yan, An 8-channel scalable EEG acquisition SoC with fully integrated patient-specific seizure classification and recording processor, in: International Solid-State Circuits Conference, 2012, pp. 164–165.

[16] C. Ling, Q. Yang, J. Wang, S. Zhang, Decision trees with minimal costs, in: Proceedings of the International Conference on Machine Learning, 2004.

[17] Z. Xu, M. Kusner, K. Weinberger, M. Chen, Cost-sensitive tree of classifiers, in: Proceedings of the International Conference on Machine Learning, 2013.

[18] C. Li, W. Zhang, C.-B. Cho, T. Li, SolarCore: Solar energy driven multi-core architecture power management, 2011 IEEE 17th International Symposium on High Performance Computer Architecture (2011) 205–216.

[19] C. Ye, Y. Zheng, S. Velipasalar, M. C. Gursoy, Energy-aware and robust task (re)assignment in embedded smart camera networks, 2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance (2013) 123–128.

[20] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection, ACM Computing Surveys 41 (3) (2009) 1–58.

[21] K. Hempstalk, E. Frank, I. Witten, One-class classification by combining density and class probability estimation, in: Machine Learning and Knowledge Discovery in Databases, Springer, Antwerp, Belgium, 2008, pp. 505–519.

[22] L. Ba, R. Caurana, Do Deep Nets Really Need to be Deep?, arXiv:1312.6184 (2013) 1–6.

[23] Y. Hu, P. C. Loizou, Subjective comparison and evaluation of speech enhancement algorithms, Speech communication 49 (7) (2007) 588–601.

[24] J. Ramírez, J. Górriz, Speech/non-speech discrimination based on contextual information integrated bispectrum LRT, Signal Processing Letters, IEEE 13 (8) (2006) 497–500.

[25] J. Thiemann, N. Ito, E. Vincent, DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments (2013).
URL `http://parole.loria.fr/DEMAND/`

[26] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, SenSay: a context-aware mobile phone, in: Proceedings of IEEE International Symposium on Wearable Computers, 2003, pp. 248–249.

[27] Y. Jiang, K. Li, R. Piedrahita, Y. Xiang, L. Tian, User-Centric Indoor Air-Quality Monitoring on Mobile Devices, AI Magazine 32 (2) (2013) 11–30.